Data Science and Analytics (IT3080)

B.Sc (Hons) in IT (IT Specialization)

Year 3

Lab Sheet 03

Clustering with KMeans Algorithm

Activity 1

Open Jupyter Notebook and create a folder named lab 04. Download the data sets for lab 04 from the courseweb.

Explore the content of the countries.csv file. The file contains Longitude and Latitude information of 241 countries.

Now create a new Notebook under Jupyter Notebook. Similar to the previous lab sheets you need to import the numpy, pandas and matplotlib.pyplot library first. Additionally add the following statements to add two more libraries.

```
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
```

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Import the data from the countries.csv file and store it in a data frame. View the content of the data using methods known to you.

Plotting the data

Use the scatter function in the matplotlib library to plot the data. For the function you need to pass the x and y axis to take as inputs. The function should be of following format.

```
matplotlib.pyplot.scatter(x, y)
```

Use the following statements to set the limits for x and y axis of the scatter plot and to view the scatter plot

Selecting the features

Next you need to extract the data you need for cluster analysis from the data frame. Note that for our purpose we do not need the country name. Therefore, create a new data frame object and assign the extracted content from the earlier data frame to the new data frame.

Extracting data from a data frame could be done using the iloc method. DataFrame.iloc[<*range*>] method should be passed with the range of data to extract. In this case, we need content of all rows but only column 1 and 2. View the content of the new data frame to see whether you got the content correctly.

Standardize the variables

The first thing we do is standardizing the variables to have **mean as 0 and standard deviation as 1**. The standardization is important since the variables have different ranges, which would have serious effect on the distance measure (i.e. Euclidean distance).

```
ss = StandardScaler()
X = pd.DataFrame(ss.fit_transform(X), columns=['column_name1', 'column_name2'])
```

Clustering

Use the following code segment to initialize clustering

```
kmeans = KMeans(<number of clusters>)
```

You can use any number as the number of clusters applicable. Next use the fit methods to perform the clustering as shown. Make sure that you input the data frame with correctly extracted values in the statement below.

kmeans.fit(<*dataframe*>)

Then, use the following statement to extract the results.

identified_clusters = kmeans.fit_predict(*<dataframe>)*

Identified_cluster is an array which contains the prediction of clusters for each country. View the content of the identified_clusters. You will note that an array of numbers is resulted.

Clustering results

Create a new data frame and assign it with a copy of the initial data frame with country name, longitude and latitude. A copy of an exisiting data frame could be created using DataFrame.copy() method. Add a new column to the newly created data frame named clusterNo and assign it the array of cluster numbers you have obtained in the previous section.

View the content of the new data frame and observe how each country is allocated to different clusters.

Plot the data in the new data frame (the one with clusterNo) on a scatter plot again, this time with colors assigned to each cluster. This could be done by adding a third parameter c=<column based on which the spots to be colored>. In this case, we want our points to be colored based on the clusterNo column. Add a forth parameter cmap='rainbow' to view the graph in more attractive colors.

Obtaining the optimal number of clusters

Use the following code segment to generate WCSS for different number of clusters with same data in our data frame.

```
wcss=[]

for i in range(<starting number>,<ending number>):

   kmeans = KMeans(i)

   kmeans.fit(<data frame>)

   wcss_iter = kmeans.inertia_

   wcss.append(wcss_iter)
```

Now plot the WCSS values you have obtained using the following code segment.

```
number_clusters = range(<starting number>,<ending number>) plt.plot(number_clusters,wcss)

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('Within-cluster Sum of Squares')
```

What do you think is the optimal number of clusters in the dataset?

Activity 2

Explore the contents of the Customers.csv in the datasets for lab4 in the courseweb. Create a new Notebook to do the following.

1. Plot age against annual income. How many clusters can you observe at a glance?

2. Check the outliers in age and annual income. Cap the outliers if any.
3. Standardize the variable values and identify the optimal clusters available on age vs annual income
4. Visualize the optimal clusters with centroids
5. Use the Kmeans algorithm to identify the optimal clusters available on age vs Spending Score and annual income vs Spending Score (Repeat the steps in 2-4)

```
import seaborn as sns
sns.set()
from sklearn.cluster import

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
data = pd.read_csv('Countries.csv')
data.head()

plt.scatter(data['Longitude'],data['Latitude'])

new_data = data.iloc[:,[1,2]]
```

create a new data frame, with only colums we need. 0 is the first colum, 1 is the second column
: selecting all rows

```
from sklearn.preprocessing import
StandardScaler

ss = StandardScaler()
new_data =
pd.DataFrame(ss.fit_transform(new_data),
columns=['Longitude', 'Latitude'])

kmeans = KMeans(4)
kmeans.fit(new_data)

identified_clusters =
kmeans.fit_predict(new_data)
identified_clusters

neww_data =data.copy()
neww_data.head()
neww_data['ClusterNo'] = identified_clusters
neww_data.head()

plt.scatter(neww_data['Longitude'],neww_data['Latitude']
,c=neww_data['ClusterNo'],cmap='rainbow')

wcss=[]

for i in range(1,11):
   kmeans = KMeans(i)
   kmeans.fit(new_data)
   wcss_iter = kmeans.inertia_
   wcss.append(wcss_iter)
```

To get better results we use less records. in the table there are 240 records

```
number_clusters = range(1,11)
plt.plot(number_clusters,wcss)
plt.title('The Elbow Method')
plt.xlable('Number of clusters')
plt.ylable('within-cluster Sum of squares')
```

from here ther is a sudden drop in the plt. from that we can understand there clusters we need is 3. prviously it was an assumption to use 4